Strategic Reasoning under Capacity-constrained Agents

Gabriel Ballot
SEIDO Lab, EDF R&D and Télécom Paris, Institut Polytechnique de Paris
Palaiseau, France
gabriel.ballot@telecom-paris.fr

Vadim Malvone
LTCI, Télécom Paris, Institut Polytechnique de Paris
Palaiseau, France
vadim.malvone@telecom-paris.fr

Jean Leneutre
LTCI, Télécom Paris, Institut Polytechnique de Paris
Palaiseau, France
jean.leneutre@telecom-paris.fr

Youssef Laarouchi
SEIDO Lab, EDF R&D
Palaiseau, France
youssef.laarouchi@edf.fr

ABSTRACT

Personality traits, experience level, or physical characteristics can affect the *capacity* (or profile) of an agent. For instance, a basketball player may be right-handed or left-handed, and these two versions cannot do the same actions. Dribbling past the player may require, first, understanding its handedness and, accordingly, execute a trick. Generally, capacities apply to systems where multiple entities can play the same role in the system, such as different client versions in protocol analysis, different robots in heterogeneous fleets, different personality traits in social structure modeling, or different attacker profiles in cybersecurity. With the capacity of other agents being unknown at the system's initialization, the hardness of imperfect information arises. Our contributions are: (*i*) introducing Capacity Alternating-time Temporal Logic (CapATL) to reason about concurrent game structure where agents are bounded to capacities, (*ii*) a model-checking algorithm for CapATL, and (*iii*) a case study of adaptive honeypot design for cyber deception.

KEYWORDS

multi-agent system verification; strategic reasoning; cybersecurity

ACM Reference Format:
Gabriel Ballot, Vadim Malvone, Jean Leneutre, and Youssef Laarouchi. 2024. Strategic Reasoning under Capacity-constrained Agents. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Engineers design increasingly complex systems and specifications often require books to be written on. It is an illusion to believe that implementations behave as specified, even after final proofreading by an expert. Researchers developed formal verification techniques to tackle this issue and rigorously prove systems' correctness. Model checking [12] is the branch of formal verification that aims to check specifications for all system computations. It relies on three components: a modeling formalism to abstract the real system, a specification formalism for expressing non-ambiguous properties, and a model-checking algorithm to verify if a given property holds on a given model.

Verification was applied to closed systems until the early 2000s. However, the need to analyze open systems appeared because of their interconnection. The model-checking community get interested in verifying multi-agent systems (MASs) that depict the interaction between different entities. Most notably, Alur *et al.* introduced Alternating-time Temporal Logic (ATL) [3] to express the strategic ability of a set of agents, called *coalition*, to ensure temporal objectives. ATL-based logics are interpreted over concurrent game structure (CGS): from a given state, each agent chooses one of its available actions, and the game progresses to a new state according to the agents' joint action. For example, the ATL property $readCmd \rightarrow \langle controller \rangle (\neg write)\ \mathcal{U}\ read$, could mean "if a `read` command arrives, the memory controller can prevent any write in the register until the read happens". The success of ATL comes from its computational simplicity because its model-checking problem is PTIME-complete. ATL paved the way for a multitude of extensions regarding epistemic [18, 22, 31, 33, 34], quantitative [2, 29], probabilistic [11, 17, 29, 30], or real-time [10, 13] objectives, as well as strategy class specifications in the description logic [28, 35, 36].

One of the most active research direction is epistemic ATL-based extensions. Each agent a gets an indistinguishability relation $\sim_a$ on states. If $s \sim_a s'$, agent a cannot tell if the system is in state s or s'. It also introduces the notion of uniform strategy, meaning agents' strategy must give the same action for indistinguishable histories. Van der Hoek and Wooldridge introduced the first version of ATEL [33]. It extends ATL with a knowledge operator $\mathcal{K}_a$ such that a knows ϕ in a state s if ϕ holds in all indistinguishable states $s' \sim_a s$. However, the strategic operator did not use uniform strategies, which was counterintuitive because agents can decide their actions according to information they cannot observe. Jamroga redefined ATEL [18] with the uniformity condition on agents' strategies as in [3]. Schobbens [31] imposed the uniformity condition also in the initial state. Finally, in 2004, Jamroga and van der Hoek defined ATOL [22] similarly to ATEL, but the coalition knowledge about the strategy is specified in the language. A fundamental result of ATL with imperfect information and memoryful strategies (depending on the whole history) is the undecidability of the model-checking problem [14]. Consequently, researchers seek workarounds to establish decidability results like bounded-memory strategies [31], natural strategies [20, 21], particular information models (static, dynamic, or recurring) [9], or approximate verification [5, 8, 15, 19].

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Contribution. This paper introduces Capacity Alternating-time Temporal Logic (CapATL) that extends ATL with the notion of agents' *capacities*. At the system's initialization, each agent chooses secretly a capacity, introducing imperfect information. This capacity determines the set of actions available for the agent. This notion is intuitive since, in various situations, a single agent can encompass different entities. For instance, in sports, the opponent may be right-handed or left-handed, and the player's actions may depend on it. When the game starts, the agents do not know the opponent's handedness, but they might identify it during the game and act consequently. More practically, we see various scenarios where agents can have different private capacities. In distributed computing, an agent may run one among different protocol versions and not declare it publicly. Finding a distributed protocol verifying good properties where protocol versions are uncertain is challenging. Conversely, CapATL could easily model it using one capacity per client version. In robotics, a heterogeneous fleet could be modeled with a capacity per type of robot. In social structure modeling, the different personality traits of agents (altruists, adventurous, selfish, *etc.*) can be capacities. In cyber security, the attacker's capacities can correspond to its resources and skills. Identifying the attacker's capacity and responding accordingly is a fundamental and challenging problem. The contributions of this paper are: (*i*) introducing CapATL, to reason about CGS with capacities, (*ii*) solving CapATL model checking with a NEXPTIME algorithm, and (*iii*) a case study where CapATL synthesizes adaptive cyber honeypots strategies.

Related work. CapATL is closely related to ATEL [31]. Indeed, CapATL can be reduced to ATEL with an exponential cost on the number of states (duplicating the CapATL structure for each combination of capacities). However, CapATL semantics considers that agents have perfect recall, hence, the reduction is into an undecidable problem [14] (ATEL with perfect recall). This emphasizes that CapATL is not just a syntactic sugar for ATEL. In contrast, CapATL highlights a new decidable fragment of ATEL with perfect recall, which does not fall into the other known decidable fragments [9].

The notion of capacity helps reasoning on the notion of strategy: an agent with a capacity constraint has a strategy to achieve a goal if this strategy uses only actions available for this capacity. Actions are also specified explicitly in AT(E)L-A [1] and ATLEA [16]: the strategic operator imposes some agents' actions for the next transition only. A more explicit strategy manipulation in the language is defined in ATLES [35] and IATL [36]. The first adds an explicit strategy commitment operator, and the second forces agents to keep their strategy when nested strategic operators have different coalitions. As opposed to CapATL these logics do not consider multiple profiles for agents. In 2014, Mogavero *et al.* defined SL [28] with more descriptive power regarding strategies because they are treated as first-order objects and are quantified independently from agents. While ATL uses a strategy quantifier for agents coalition $\langle Y \rangle$ ("there exists a strategy for agents in Y"), SL separates the strategy quantification $\exists s$ ("there exists a strategy s") and the agent strategy assignment (x, s) ("agent x uses strategy s"). As such, SL can give the same strategy to several agents or specify Nash equilibriums. SL model-checking problem is non-elementary with respect to the formula size (and PTIME-complete with respect to the game structure). Several fragments have been studied to reduce the complexity [7, 28]. However, none of these logics reason about strategies regarding the actions performed. So, CapATL tackles a different aspect.

Back in 1993, van der Hoek *et al.* defined a logic of capabilities [32] where, in each state, each agent is able to perform a given set of actions. The logic has an operator $A_a\alpha$ which is true if the agent a can perform the action α in the current state. However, there is no notion of different capacities for a single agent: in CapATL, the agents must commit to performing the same set of actions for the whole computation. This dynamic aspect is not present in the logic of capabilities. Lesperance *et al.* also proposed a formal framework to reason about "knowing how to do an action" [24]. Once again, it does not yield the dynamic aspect of CapATL.

Outline. The rest of this paper is organized as follows: our game structure and logic are introduced respectively in Sections 2 and 3, the model checking is established in Section 4, a case study is developed in Section 5, and Section 6 concludes this paper.

2 CGS WITH CAPACITIES

This section defines the system's modeling formalism and the possible computations on that system. To follow the tradition of post-2002 ATL-based logics, we rely on the general framework of CGS, with two differences. First, we add the notion of capacities in the structure. Second, we name actions explicitly because they are particularly meaningful with respect to capacities.

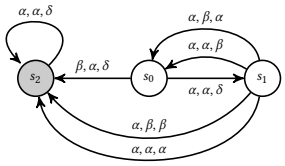
Given a set X, $\mathcal{P}(X)$ denotes the set of subsets of X. We write $f : X \rightharpoonup Y$ (resp. $g : X \to Y$) to introduce a partial function f (resp. application g) from a set X to a set Y, and $\mathrm{dom}(f)$ denotes the domain of f. The set of positive integers is denoted by $\mathbb{N}$. Definition 2.1 defines the *Capacity Concurrent Game Structure (CapCGS)*.

Definition 2.1 (Capacity Concurrent Game Structure). A CapCGS $\langle \mathrm{Agt}, \mathrm{Cap}, \mathrm{St}, \Pi, \pi, \mathrm{Act}, \Gamma, \gamma, d, o \rangle$ is a structure with the following attributes: a set of n agents $\mathrm{Agt} = \{1, \ldots, n\}$, a finite set of capacities Cap, a finite set of states St, a finite set of atomic propositions Π, a labeling function $\pi : \mathrm{St} \to \mathcal{P}(\Pi)$, a finite set of actions Act, a function $\Gamma : \mathrm{Agt} \to \mathcal{P}(\mathrm{Cap})$ that assigns a subset of capacities to each agent, a function $\gamma : \mathrm{Cap} \to \mathcal{P}(\mathrm{Act})$ that assigns a subset of actions to each capacity, a protocol function $d : \mathrm{Agt} \times \mathrm{St} \to \mathcal{P}(\mathrm{Act})$ where $d(a, s)$ is the set of actions available for the agent $a \in \mathrm{Agt}$ in the state $s \in \mathrm{St}$ and verifies $d(a, s) \subseteq \bigcup_{c \in \Gamma(a)} \gamma(c)$ and $d(a, s) \cap \gamma(c) \neq \emptyset$ for all $c \in \Gamma(a)$, and a partial transition function $o : \mathrm{St} \times \mathrm{Act}^n \rightharpoonup \mathrm{St}$ defined for all $(s, \alpha_1, \ldots, \alpha_n)$ verifying $\alpha_a \in d(a, s)$ for all $a \in \mathrm{Agt}$.

The restriction on the protocol function imposes that every agent, whatever its capacity, has at least one available action, and a transition exists for all combinations of agents' available actions.

Example 2.2. Figure 1 is an example of CapCGS with three agents $\mathrm{Agt} = \{1, 2, 3\}$. Agents 1 and 2 have only one capacity $\Gamma(1) = \Gamma(2) = \{c\}$ and the actions possible for c are $\gamma(c) = \{\alpha, \beta\}$. Agent 3 has two possible capacities $\Gamma(3) = \{c_1, c_2\}$ and capacity c_1 (resp. c_2) let the agent do actions α and δ (resp. β and δ), *i.e.*, $\gamma(c_1) = \{\alpha, \delta\}$ and $\gamma(c_2) = \{\beta, \delta\}$. We have $\Pi = \{gray\}$ and $gray$ holds only in s_2, *i.e.*, $\pi(s_0) = \pi(s_1) = \emptyset$ and $\pi(s_2) = \{gray\}$.

In the rest of this paper, we consider a general *CapCGS* $\mathcal{S} = \langle \mathrm{Agt}, \mathrm{Cap}, \mathrm{St}, \Pi, \pi, \mathrm{Act}, \Gamma, \gamma, d, o \rangle$ with n agents. Intuitively, during



Agent	Capacity	Actions
1	c	$\{\alpha, \beta\}$
2	c	$\{\alpha, \beta\}$
3	c_1	$\{\alpha, \delta\}$
	c_2	$\{\beta, \delta\}$

Figure 1: A simple CapCGS with three agents.

a play, each agent $a \in \text{Agt}$ will secretly choose one of its capacities $c \in \Gamma(a)$, meaning that a can use only the actions in $\gamma(c)$. An interesting question for the verification of complex systems with capacity constraints (and we will formalize it in Section 3) is whether a coalition has a strategy to guarantee temporal and epistemic properties, including identifying the capacity of other agents. First, we remind some definitions that apply to CGS and are adapted to CapCGS. A *path* describes the possible realizations of the game. It contains information about the succession of states and actions of all the agents. Formally, a path ρ is a possibly infinite word $\rho = s_1\vec{\alpha}_1 s_2\vec{\alpha}_2 \dots$ where $\vec{\alpha}_i = (\alpha_i^1, \dots, \alpha_i^n) \in \text{Act}^n$ is the agents' joint action at step i. It must satisfy for all i, $s_{i+1} = o(s_i, \alpha_i^1, \dots, \alpha_i^n)$. If a path is finite, it ends with a state. The set of paths, finite paths, and infinite paths are respectively denoted by Paths, $\text{Paths}^{<\omega}$, and Paths^{ω}. We denote by $|\rho|$ the number of states in ρ (ρ has $|\rho| - 1$ joint actions), and $|\rho| = \omega$ if ρ is infinite. We denote ρ's prefix by $\rho_{\leq i} = s_1\vec{\alpha}_1 \dots \vec{\alpha}_{i-1}s_i$ (which has $2i - 1$ symbols) and, if $|\rho| < \omega$, we denote ρ's last state by $\text{last}(\rho)$. We access ρ's i^{th} state by $\rho[i] = s_i$ and, given a joint action $\vec{\alpha} = (\alpha_1, \dots, \alpha_n) \in \text{Act}^n$, we denote by $\vec{\alpha}[a] = \alpha_a$ the action of agent a.

Example 2.3. In the setting of Example 2.2, we can define the paths $\rho = s_1(\alpha, \alpha, \beta)s_0(\alpha, \alpha, \delta)s_1(\alpha, \alpha, \beta)s_0$ and, changing the last transition, $\eta = s_1(\alpha, \alpha, \beta)s_0(\alpha, \alpha, \delta)s_1(\alpha, \beta, \alpha)s_0$.

We consider agents cannot observe other agents' actions, but only the sequence of paths and their actions. Definition 2.4 formalizes it with an equivalence relation over paths for each agent, called indistinguishability relation.

Definition 2.4 (Indistinguishability). Two paths $\rho = s_1\vec{\alpha}_1 s_2\vec{\alpha}_2 \dots$ and $\eta = q_1\vec{\beta}q_2\vec{\beta}_2 \dots$ are indistinguishable for an agent $a \in \text{Agt}$, denoted by $\rho \sim_a \eta$, iff (*i*) $|\rho| = |\eta|$, (*ii*) for all $i \in \{1, \dots, |\rho|\}$, $s_i = q_i$, and (*iii*) for all $i \in \{1, \dots, |\rho| - 1\}$, $\vec{\alpha}_i[a] = \vec{\beta}_i[a]$.

A (memoryful) *strategy* is a function $s : \text{Paths}^{<\omega} \to \text{Act}$ that maps each finite path to an action. A strategy s for an agent $a \in \text{Agt}$ is called uniform (for $\sim_a$) if, for all finite paths ρ and η, $\rho \sim_a \eta$ implies $s(\rho) = s(\eta)$. We will use assignment functions similarly to [28] to assign a strategy or a capacity to an agent.

Definition 2.5 (Memoryful uniform strategy assignment). A memoryful uniform strategy assignment is a partial function $\sigma : \text{Agt} \rightharpoonup (\text{Paths}^{<\omega} \to \text{Act})$ that assigns memoryful uniform strategies to agents. For all $a \in \text{dom}(\sigma)$, there must exist a capacity $c \in \Gamma(a)$ such that, for all $\rho \in \text{Paths}^{<\omega}$, $\sigma(a)(\rho) \in d(a, \text{last}(\rho)) \cap \gamma(c)$.

Definition 2.6 (Capacity assignment). A capacity assignment is a partial function $\kappa : \text{Agt} \rightharpoonup \text{Cap}$ that assigns capacities to agents, *s.t.*, for an agent $a \in \text{Agt}$, we have $\kappa(a) \in \Gamma(a)$.

In ATL, whenever a memoryful strategy satisfies a property ϕ_{ATL}, then a memoryless strategy (mapping only states to the actions) also exists to satisfy ϕ_{ATL}. As such, ATL reasoning needs only to deal with memoryless strategies. However, this is not true in CapATL (*cf.*, Theorem 3.4), and we have to reason about the memoryful class of strategies. This emphasizes that CapATL analysis requires more involved tools than ATL analysis. For the rest of this paper, all strategies are memoryful and uniform by default. We say that a (capacity or strategy) assignment is *complete* if its domain is Agt.

Agents cannot change capacities on a path ρ in a CapCGS. Consequently, we can rule out the complete capacity assignments that do not allow the agent to use some actions from ρ. We formalize this in the notion of ρ-compatible assignments.

Definition 2.7 (Path compatible assignments). Given a path $\rho = s_1\vec{\alpha}_1 \dots \vec{\alpha}_{k-1}s_k$, we let $C(\rho)$ denote the set of possible complete capacity assignments that may bring about ρ. We have $\kappa \in C(\rho)$ iff for all $i \in \{1, \dots, k\}$ and $a \in \text{Agt}$, $\vec{\alpha}_i[a] \in \gamma(\kappa(a))$.

Example 2.8. Using the paths ρ and η defined in Example 2.3, on the one hand, we have $C(\rho) = \{\kappa\}$ where $\kappa(1) = c$, $\kappa(2) = c$, and $\kappa(3) = c_2$. This capacity assignment is the only one that could give the path ρ. On the other hand, $C(\eta) = \emptyset$ because agent 3 has no capacity that allow actions α and β in the same path.

Let $\rho = s_1\vec{\alpha}_1 s_2\vec{\alpha}_2 \dots$ be a path. Notice that we can decompose $C(\rho) = \bigcap_{i \in \{1,\dots,|\rho|-1\}} C(s_i\vec{\alpha}_i s_{i+1})$. This result shows that the multiplicity of a same transition in a path and the transition order do not matter to know what are the compatible capacity assignments. We can now define the outcomes of a strategy assignment from a state. It returns the extending paths respecting a capacity assignment and the input strategy assignment.

Definition 2.9 (Outcomes). Let $s \in \text{St}$ and σ be a strategy assignment for a coalition $Y \subseteq \text{Agt}$. The set of *outcomes* $\text{Out}(s, \sigma) \subseteq \text{Paths}^{\omega}$ is a set of infinite paths $\eta = s_1\vec{\alpha}_1 s_2\vec{\alpha}_2 \dots$ such that $s_1 = s$, $C(\eta) \neq \emptyset$, and, for all $i \in \mathbb{N}$ and $a \in Y$, $\vec{\alpha}_i[a] = \sigma(a)(s_1\vec{\alpha}_1 \dots \vec{\alpha}_{i-1}s_i)$.

3 CAPACITY ATL

This section introduces CapATL to reason about the strategic, epistemic, and temporal properties of CapCGS. Subsection 3.1 defines CapATL's syntax and Subsection 3.2 gives its semantics.

3.1 Syntax

CapATL is an extension of ATL [3] that allows to reason about capacities. It can express properties like "Can a coalition of agents guess other agents' capacities?". It adds a knowledge operator to ATL that contains a capacity assignment formula. This subformula is a propositional formula that characterizes a set of complete capacity assignments.

Definition 3.1 (CapATL syntax). The following grammar defines a CapATL formula ϕ:

$$\phi ::= \ell \mid \mathcal{K}^a_{\text{cap}}(\varphi) \mid \neg\phi \mid \phi \wedge \phi \mid \langle Y \rangle \psi$$
$$\psi ::= \mathcal{X}\,\phi \mid \phi\,\mathcal{U}\,\phi \mid \phi\,\mathcal{R}\,\phi$$
$$\varphi ::= a \mapsto c \mid \neg\varphi \mid \varphi \wedge \varphi$$

where $\ell \in \Pi$ is an atomic proposition, $Y \subseteq \text{Agt}$ is an agent coalition, $a \in \text{Agt}$ is an agent, and $c \in \text{Cap}$ is a capacity.

As in ATL, $\langle\cdot\rangle$ is the *strategic operator*, and $\langle Y\rangle\psi$ means that Y has a strategy to enforce ψ (called *temporal formula*) whatever the actions of the other agents. The strategic operator is immediately followed by a *temporal operator*, either $\mathcal{X}$ for "next", $\mathcal{U}$ for "until", or $\mathcal{R}$ for "release" (the dual of $\mathcal{U}$). Finally, $\mathcal{K}^a_{\text{cap}}$ is the *knowledge operator* and $\mathcal{K}^a_{\text{cap}}(\varphi)$ means that agent a knows that the capacity assignment verifies φ where φ is called *capacity assignment formula*: it characterizes a set of capacity assignments. For example, $a \mapsto c_1 \wedge (b \mapsto c_2 \vee b \mapsto c_3)$—where $\vee$ is defined as usual—characterizes the set of complete capacity assignments κ verifying $\kappa(a) = c_1$ and $\kappa(b) \in \{c_2, c_3\}$. The original syntax for ATL defines the "globally" operator $\mathcal{G}\,\phi$ instead of the "release" operator $\phi\,\mathcal{R}\,\phi$. But as noticed by [23], in ATL, $\mathcal{R}$ cannot be inferred from $\mathcal{G}$ and $\mathcal{U}$ as in CTL, or from the dual of $\mathcal{U}$ because the negation of temporal formula is not defined. It is noteworthy that our knowledge operator $\mathcal{K}^a_{\text{cap}}$ deals with a's knowledge about agents' capacities, while the usual knowledge operator in epistemic logics handles properties of the model (*e.g.*, "Do agents know that the current state has some atomic proposition ℓ?").

Syntactic sugar. For ϑ_1 and ϑ_2 two CapATL or capacity assignment formulae, the disjunction $\vartheta_1 \vee \vartheta_2$ and implication $\vartheta_1 \rightarrow \vartheta_2$ are defined as usual, as well as $\top$ and $\bot$. The dual of the strategic operator is denoted by $[Y]\psi$, meaning "Y cannot avoid ψ", and the dual $\mathcal{M}^a_{\text{cap}}(\varphi) = \neg\mathcal{K}^a_{\text{cap}}(\neg\varphi)$ means "the capacity assignment *might* verify φ from agent a's point of view". The *globally* operator is $\mathcal{G}\,\phi = \bot\,\mathcal{R}\,\phi$, and the *eventually* operator is $\mathcal{F}\phi = \top\,\mathcal{U}\,\phi$. The strategic operator is implicitly quantified with an existential quantifier for the capacity of the strategic coalition (*cf.*, Definition 3.2). To refine this behavior, a formula of the form $\langle\varphi\rangle\psi$ where φ is a capacity assignment formula involving the agents forming a coalition Y means that "Y can enforce ψ using an assignment verifying φ". For instance, $\langle a \mapsto c_1 \wedge \neg b \mapsto c_2\rangle\psi$ means that the coalition $\{a, b\}$ has a strategy to enforce ψ, a can use c_1 for this strategy, and b can use a capacity different from c_2 for this strategy. The formula $\langle\varphi\rangle\psi$ can be derived from the syntax as follows. Let Y denote the set of agents occurring in φ and $M = \bigwedge_{a\in Y} \mathcal{M}^a_{\text{cap}}(\varphi)$ which is true iff agents in Y can use capacities verifying φ. We have $\langle\varphi\rangle\,\mathcal{X}\,\phi = \langle Y\rangle\,\mathcal{X}(\phi \wedge M)$, $\langle\varphi\rangle\phi_1\,\mathcal{U}\,\phi_2 = \langle Y\rangle\phi_1\,\mathcal{U}\,(\phi_2 \wedge M)$, and $\langle\varphi\rangle\phi_1\,\mathcal{R}\,\phi_2 = \langle Y\rangle\phi_1\,\mathcal{R}\,(\phi_2 \wedge M)$. Finally, for two agents a and b, we write $\mathcal{K}^a_{\text{cap}}(b) = \bigvee_{c\in\Gamma(b)} \mathcal{K}^a_{\text{cap}}(b \mapsto c)$ to simplify a formula meaning that agent a knows agent b's capacity.

3.2 Semantics

CapATL semantics is formalized through a satisfaction relation for an infinite path, an index of this path, and a capacity assignment. Note that ATL uses paths containing only states instead of paths with states and joint actions because the actions between states do not matter. However, in CapATL, having the actions is essential to determine what agents know.

Definition 3.2 (CapATL semantics). Let ρ be an infinite path, $i > 0$ be an index, κ be a complete capacity assignment, ℓ be an atomic proposition, a be an agent, Y be a coalition of agents, (ϕ, ϕ_1, ϕ_2) be three CapATL formulae, ψ be a temporal formula, and $(\vartheta, \vartheta_1, \vartheta_2)$ be three CapATL or capacity assignment formulae. CapATL semantics is defined through the following satisfaction relation:

- $(\rho, i, \kappa) \models \ell$ iff $\ell \in \pi(\rho[i])$,
- $(\rho, i, \kappa) \models a \mapsto c$ iff $\kappa(a) = c$,
- $(\rho, i, \kappa) \models \mathcal{K}^a_{\text{cap}}(\varphi)$ iff for all $\eta \sim_a \rho$, for all $\kappa' \in C(\eta_{\leq i})$, we have $(\eta, i, \kappa') \models \varphi$,
- $(\rho, i, \kappa) \models \neg\vartheta$ iff $(\rho, i, \kappa) \not\models \vartheta$,
- $(\rho, i, \kappa) \models \vartheta_1 \wedge \vartheta_2$ iff $(\rho, i, \kappa) \models \vartheta_1$ and $(\rho, i, \kappa) \models \vartheta_2$,
- $(\rho, i, \kappa) \models \langle Y\rangle\psi$, iff there is a strategy assignment σ for Y—called winning strategy—such that, for all outcomes $\eta \in \text{Out}(\rho[i], \sigma)$, we have $(\eta, 1, \kappa) \models \psi$,
- $(\rho, i, \kappa) \models \mathcal{X}\,\phi$ iff $(\rho, i + 1, \kappa) \models \phi$,
- $(\rho, i, \kappa) \models \phi_1\,\mathcal{U}\,\phi_2$ iff there exists $j \geq i$, *s.t.*, we have $(\rho, j, \kappa) \models \phi_2$ and for all k where $i \leq k < j$, we have $(\rho, k, \kappa) \models \phi_1$,
- $(\rho, i, \kappa) \models \phi_1\,\mathcal{R}\,\phi_2$ iff either (i) for all $j \geq i$, we have $(\rho, j, \kappa) \models \phi_2$, or (ii) there exists $j \geq i$, *s.t.*, $(\rho, j, \kappa) \models \phi_1 \wedge \phi_2$ and for all k where $i \leq k < j$, we have $(\rho, k, \kappa) \models \phi_2$.

Notice that, for a CapATL formula ϕ, $(\rho, i, \kappa) \models \phi$ depends only on $\rho_{\leq i}$ and ϕ, so we may write $\rho_{\leq i} \models \phi$ (in particular $s \models \phi$ has a meaning). Moreover, for a capacity assignment formula φ, $(\rho, i, \kappa) \models \varphi$ depends only on κ and φ, so we may write $\kappa \models \varphi$. Besides, the semantics of $\langle Y\rangle\psi$ has an existential quantification over the strategy assignments for Y and a universal quantification over the outcomes. It implies an implicit existential (resp. universal) quantification over Y (resp. Agt $\setminus$ Y) capacity assignments compatible with future actions in each outcome. Agents can change their capacity for each nested strategic operator. Another approach (not discussed) could consist in quantifying over capacity assignments only once.

Example 3.3. Consider the system from Example 2.2 illustrated in Figure 1. The formula $\phi = \langle 1\rangle\mathcal{F}(gray \wedge \mathcal{K}^2_{\text{cap}}(3))$—expanded as $\langle 1\rangle\top\mathcal{U}\left(gray \wedge (\mathcal{K}^2_{\text{cap}}(3 \mapsto c_1) \vee \mathcal{K}^2_{\text{cap}}(3 \mapsto c_2))\right)$—is true iff agent 1 can enforce that all paths eventually go in a state labelled $gray$ and agent 2 knows whether agent 3 has capacity c_1 or c_2. Notice that $\mathcal{K}^2_{\text{cap}}(3 \mapsto c_1 \vee 3 \mapsto c_2)$ semantically differs from $\mathcal{K}^2_{\text{cap}}(3 \mapsto c_1) \vee \mathcal{K}^2_{\text{cap}}(3 \mapsto c_2)$. Indeed, the first is always true because agent 3 has its capacity among $\{c_1, c_2\}$, while the second means that agent 2 knows whether 3 has c_1 or c_2. We have $s_0 \models \phi$. Indeed, agent 1 can do α in s_0 and in s_1. Then, if the computation reaches s_2 then agent 2 knows agent 3's capacity (if 2 did α then 3 did α too, so 3 must have the capacity c_1, and similarly if 2 did β). As s_2 has the $gray$ property, these paths are winning. Moreover, if after being in s_1 the computation goes back to s_0, then agent 2 also knows agent 3's capacity. Agent 1 can then do action β to reach state s_2.

Theorem 3.4 establishes that agents with memory are strictly more powerful than agents without memory for CapATL objectives. This seems intuitive, but it is not true for ATL objectives. It stresses out that CapATL is way harder than ATL.

Theorem 3.4. *CapATL with memoryful strategies is not equivalent to CapATL with memoryless strategies.*

Proof. Notice that the winning strategy for $\langle 1\rangle\mathcal{F}(gray \wedge \mathcal{K}^2_{\text{cap}}(3))$ in Example 3.3 is memoryful, and any memoryless strategy could not be a winning strategy. Indeed, if agent 1 always uses α_1 (resp. α_2) in s_0, 1 cannot guarantee $\mathcal{F}(gray)$ (resp. $\mathcal{F}(\mathcal{K}^2_{\text{cap}}(3))$). □

4 MODEL-CHECKING

This section provides the decidability of the model-checking problem for CapATL. First, we prove agents do not need to recall all the past to have a winning strategy. The following explains what suffices to recall. For two states $s, s' \in \mathrm{St}$, an action $\alpha \in \mathrm{Act}$, and an agent $a \in \mathrm{Agt}$, let $T(s\alpha s', a)$ denote the number of transition from s to s' in $\mathcal{S}$ where a does α. The memory of an agent is defined as a word $m = Q_1 t_1 Q_2 t_2 \dots Q_k$, where, for all i, $Q_i \subseteq \mathrm{St}$ and $t_i = s_i \alpha_i s_i'$, with $s_i, s_i' \in \mathrm{St}$ and $\alpha_i \in \mathrm{Act}$. We denote by $\mathrm{cnt}(s\alpha s', m)$ the number of occurrences of $s\alpha s'$ in m. For an agent a, we denote by $m_a(\rho)$ the memory that a keeps from the finite path ρ. If $\rho = s \in \mathrm{St}$, then $m_a(\rho) = \{s\}$. Inductively, for all $\rho \in \mathrm{Paths}^{<\omega}$ where s denotes $\mathrm{last}(\rho)$, for all $\vec{\alpha} \in \mathrm{Act}^n$ and $s' \in \mathrm{St}$,

$$m_a(\rho\vec{\alpha}s') = \begin{cases} m_a(\rho)s\vec{\alpha}[a]s'\{s'\} & \text{if } \mathrm{cnt}(s\vec{\alpha}[a]s', m_a(\rho)) < \\ & T(s\vec{\alpha}[a]s', a), \\ m_a(\rho) \uplus s' & \text{otherwise.} \end{cases} \quad (1)$$

where $(Q_1 t_1 \dots t_{k-1} Q_k) \uplus s' = Q_1 t_1 \dots t_{k-1}(Q_k \cup \{s'\})$. The memory $m_a(\rho)$ extracts the transitions $s\alpha s'$ in ρ from a's point of view (only a's action α) until they appear more than $T(s\alpha s', a)$ times. The memory also keeps track of the set of states visited by transitions that appeared more than $T(s\alpha s', a)$ times in the past. Intuitively, the goal is to extract the least information characterizing the set of subformulae that were satisfied along the path. The threshold $T(s\alpha s', a)$ ensures that a transition where a redoes α but other agents can change their actions (and increase their knowledge) is recorded and ensures synchronicity for agents in a coalition.

Example 4.1. In the CapCGS from Figure 1, we have $T(s_0\alpha s_1, 1) = 1$ and $T(s_1\alpha s_0, 1) = 2$ because there is 1 transition from s_0 to s_1 where agent 1 uses action α, and 2 from s_1 to s_0. Now, let ρ be the path $s_0(\alpha,\alpha,\delta)s_1(\alpha,\alpha,\beta)s_0(\alpha,\alpha,\delta)s_1(\alpha,\beta,\alpha)s_0(\alpha,\alpha,\delta)s_1(\alpha,\beta,\alpha)s_0$. We have $m_2(\rho) = \{s_0\}s_0\alpha s_1\{s_1\}s_1\alpha s_0\{s_0, s_1\}s_1\beta s_0\{s_0, s_1\}$.

We consider the partial order between memories as $m \leq m'$ if m is a prefix of m' or if $m \uplus s_1 \uplus s_2 \cdots = m'$ for some states $s_1, s_2 \dots$. Notice that, for all $a \in \mathrm{Agt}$, m_a is increasing for $\leq$ with respect to the prefix partial order on paths. The size (number of symbols) of $m_a(\rho)$ is bounded by $2^{|\mathrm{St}|}3|o|! = O(2^{(|\mathrm{St}||o|^2)})$, where $|\mathrm{St}|$ and $|o|$ are respectively the number of states and transitions in $\mathcal{S}$. Let ρ and η be two finite paths and $a \in \mathrm{Agt}$. We define the *weak memory* equivalence relation $\approx_a^{\mathsf{m}} \subseteq \mathrm{Paths}^{<\omega} \times \mathrm{Paths}^{<\omega}$ such that $\rho \approx_a^{\mathsf{m}} \eta$ iff $m_a(\rho) = m_a(\eta)$. In addition, we define the *strong memory* equivalence $\approx_a^{\mathsf{M}} \subseteq \mathrm{Paths}^{<\omega} \times \mathrm{Paths}^{<\omega}$ such that $\rho \approx_a^{\mathsf{M}} \eta$ iff $\rho \approx_a^{\mathsf{m}} \eta$ and $\mathrm{last}(\rho) = \mathrm{last}(\eta)$.

Example 4.2. In Figure 1, let $\tau = s_0(\alpha,\alpha,\delta)s_1(\alpha,\alpha,\beta)$, and the paths $\eta = \tau\tau s_0(\alpha,\alpha,\delta)s_1$ and $\mu = \tau\tau\tau s_0$. We have $\eta \approx_1^{\mathsf{m}} \mu$ because $m_1(\eta) = m_1(\mu) = \{s_0\}s_0\alpha s_1\{s_1\}s_1\alpha s_0\{s_0, s_1\}s_1\alpha s_0\{s_0, s_1\}$. However, $\mathrm{last}(\eta) \neq \mathrm{last}(\mu)$ so $\eta \not\approx_1^{\mathsf{M}} \mu$.

We define memory-bounded strategy assignments that assign uniform strategies with respect to the strong memory equivalence.

Definition 4.3 (Memory-bounded strategy assignment). A memory-bounded strategy assignment σ for Y is a strategy assignment such that, for all $a \in Y$, for all finite paths ρ and η, if $\rho \approx_a^{\mathsf{M}} \eta$ then $\sigma(a)(\rho) = \sigma(a)(\eta)$.

Notice that $\sim_a \subseteq \approx_a^{\mathsf{M}} \subseteq \approx_a^{\mathsf{m}}$, so a memory-bounded strategy assignment is also uniform with respect to $\sim_a$. For a path ρ and an equivalence relation $\sim$, we let $[\![\rho]\!]_\sim$ denote the equivalence class of ρ for $\sim$. The number of equivalence classes for $\approx_a^{\mathsf{m}}$ and $\approx_a^{\mathsf{M}}$ are bounded by $2^{(|\mathrm{St}||o|^2)}$ and $|\mathrm{St}|2^{(|\mathrm{St}||o|^2)}$, respectively. Theorem 4.4 shows that agents do not need to recall all the history to win a game. This is an essential result to prove CapATL decidability.

Theorem 4.4. *Let ψ be a CapATL temporal formula, $Y \subseteq \mathrm{Agt}$ an agent coalition, ρ an infinite path, i an index, and κ a complete capacity assignment. We have $(\rho, i, \kappa) \models \langle Y\rangle\psi$ if and only if, there exists a memory-bounded winning strategy assignment for Y.*

Proof. In this proof, we extend the satisfiability for finite path, denoted by $\models_f$. For $\rho \in \mathrm{Paths}^{<\omega}$ where $k = |\eta|$, for $i \in \{1, \dots, k\}$, for ϕ_1 and ϕ_2 two CapATL formulae, and for a complete capacity assignment κ, we let $(\rho, i, \kappa) \models_f \phi_1$ iff any infinite extension η of ρ verifies $(\eta, i, \kappa) \models \phi_1$. We let $(\rho, i, \kappa) \models_f \phi_1 \,\mathcal{U}\, \phi_2$ iff $(\rho, l, \kappa) \models_f \phi_2$ at an index $l \in \{i, \dots, k\}$ and $(\rho, j, \kappa) \models_f \phi_1$ for all $j \in \{i, \dots, l-1\}$. Similarly, we let $(\rho, i, \kappa) \models_f \phi_1 \,\mathcal{R}\, \phi_2$ if either $(\rho, j, \kappa) \models_f \phi_2$ for all $j \in \{i, \dots, k\}$, or $(\rho, l, \kappa) \models_f \phi_1 \wedge \phi_2$ for some $l \in \{i, \dots, k\}$ and $(\rho, j, \kappa) \models_f \phi_2$ for all $j \in \{i, \dots, l\}$. We extend the definition of memory for the coalition $Y = \{a, \dots, b\}$ where $a < \cdots < b$. Formally, for $s, s' \in \mathrm{St}$ and $\alpha_a, \dots, \alpha_b \in \mathrm{Act}$, we define $T(s\alpha_a \dots \alpha_b s', Y)$ the number of transition in $\mathcal{S}$ where all agents $a' \in Y$ does action $\alpha_{a'}$. We define the memory of Y, *s.t.*, for $\rho \in \mathrm{Paths}^{<\omega}$, $s = \mathrm{last}(\rho)$, $s' \in \mathrm{St}$, and $\vec{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathrm{Act}^n$:

$$m_Y(\rho\vec{\alpha}s') = \begin{cases} m_Y(\rho)s\alpha_a \dots \alpha_b s'\{s'\} & \text{if } \mathrm{cnt}(s\alpha_a \dots \alpha_b s', m_Y(\rho)) < \\ & T(s\alpha_a \dots \alpha_b s', Y), \\ m_Y(\rho) \uplus s' & \text{otherwise.} \end{cases}$$

and $m_Y(\rho) = \{s\}$ if $\rho = s$. For $\rho, \eta \in \mathrm{Paths}^{<\omega}$, we define the relations $\approx_Y^{\mathsf{m}}$ and $\approx_Y^{\mathsf{M}} \subseteq \mathrm{Paths}^{<\omega} \times \mathrm{Paths}^{<\omega}$, such that $\rho \approx_Y^{\mathsf{m}} \eta$ iff $m_Y(\rho) = m_Y(\eta)$, and $\rho \approx_Y^{\mathsf{M}} \eta$ iff $m_Y(\rho) = m_Y(\eta)$ and $\mathrm{last}(\rho) = \mathrm{last}(\eta)$. The partial order on $\approx_Y^{\mathsf{m}}$'s equivalence classes is defined as follows: $[\![\rho]\!]_{\approx_Y^{\mathsf{m}}} \leq [\![\eta]\!]_{\approx_Y^{\mathsf{m}}}$ iff $m_Y(\rho) \leq m_Y(\eta)$ (which is independent of the representatives). We define a *partial-history strategy assignment* $\sigma_f : Y \to (\mathrm{Paths}^{<\omega} \rightharpoonup \mathrm{Act})$ that gives partial strategies for each agent in Y. However, there must be $P = \bigcup_{i \in \{1,\dots,k\}} P_i$, the union of some equivalence classes $P_1, \dots, P_k$ for $\approx_Y^{\mathsf{M}}$, such that, for all $a \in Y$, $\mathrm{dom}(\sigma_f(a)) = P$. A partial-history strategy assignment is $\approx_Y^{\mathsf{M}}$-uniform. The outcomes of a partial-history strategy assignment σ_f for a coalition Y from a finite path $\rho = s_1\vec{\alpha}_1 s_2 \dots \vec{\alpha}_{k-1} s_k$, denoted by $\mathrm{Out}_f(\rho, \sigma_f)$, is the set of finite or infinite paths $\eta = \rho\vec{\alpha}_k s_{k+1} \dots$ such that, for all $j \in \mathbb{N}$ with $k \leq j \leq |\eta| - 1$ and all $a \in Y$, $\sigma_f(a)$ is defined for $s_1\vec{\alpha}_1 \dots \vec{\alpha}_{j-1} s_j$ and $\vec{\alpha}_i[a] = \sigma_f(a)(s_1\vec{\alpha}_1 \dots \vec{\alpha}_{j-1} s_j)$. If $l = |\eta| < \omega$, then $\sigma_f(a)$ is undefined for $s_1\vec{\alpha}_1 \dots \vec{\alpha}_{l-1} s_l$ for all agents $a \in Y$. It is important to notice that $\mathrm{Out}_f(\rho, \sigma_f)$ gives outcomes that extend ρ. Finally, for a finite path ρ and a path η where $\eta[1] = \mathrm{last}(\rho)$, we let $\rho \cdot \eta$ denote the concatenation of ρ and η without its first state.

We can start the proof of the theorem. One direction is immediate, because a memory-bounded strategy is a uniform strategy. Conversely, suppose $(\rho, i, \kappa) \models \langle Y\rangle\psi$ with a winning strategy assignment σ (for the rest of the proof, ρ, i, and σ will always refer to this). We want to build a memory-bounded winning strategy assignment σ'. The case $\langle Y\rangle \mathcal{X}\, \phi$ does not require memory since

the agents take only one action. Suppose the formula is $\langle Y\rangle\phi_1 \mathcal{U}\, \phi_2$. Let Ω denote the set of finite paths strongly memory equivalent to a winning outcome prefix, *i.e.*, the finite paths $\eta \in \mathrm{Paths}^{<\omega}$ such that there is $\mu \in \mathrm{Out}(\rho[i], \sigma)$ and $j \in \mathbb{N}$ verifying $\mu_{\leq j} \approx_Y^{\mathsf{M}} \eta$. Let Θ be the set of winning outcomes prefixes that have not satisfied ϕ_2 yet, *i.e.*, $\Theta = \{\eta_{\leq j} \mid \eta \in \mathrm{Out}(\rho[i], \sigma) \wedge j \in \mathbb{N} \wedge (\eta, 1, \kappa) \models_f \mathcal{G}(\phi_1 \wedge \neg\phi_2)\}$ (recall that the choice of the complete capacity assignment κ does not influence the satisfaction of the formula). For a partial-history strategy assignment σ_f, let $\mathcal{H}_{\sigma_f}$ denote the hypothesis "for all $\eta \in \Omega$ and $\mu \in \mathrm{Out}_f(\eta, \sigma_f)$, either: *(i)* $\mu = \eta$, *i.e.*, σ_f is not defined for η, *(ii)* $(\mu, 1, \kappa) \models_{(f)} \phi_1 \mathcal{U}\, \phi_2$ where $\models_{(f)}$ is $\models$ or $\models_f$ depending on $|\mu|$, or *(iii)* $\mu \in \Omega$ and $[\![\eta]\!]_{\approx_Y^{\mathsf{m}}} < [\![\mu]\!]_{\approx_Y^{\mathsf{m}}}$".

The partial-history strategy assignment $\sigma_\emptyset$ for Y, defined for the empty set of histories, verifies $\mathcal{H}_{\sigma_\emptyset}$ because of the case *(i)* of $\mathcal{H}_{\sigma_\emptyset}$.

Now, suppose $\mathcal{H}_{\sigma_f}$ holds for some partial-history strategy assignment for Y that is not defined for all histories. The following builds a partial-history strategy assignment σ'_f with strictly larger history domain and such that $\mathcal{H}_{\sigma'_f}$ holds. If, for all $\eta \in \Omega$, all outcomes $\mu \in \mathrm{Out}_f(\eta, \sigma_f)$ are such that we are in case *(ii)* of $\mathcal{H}_{\sigma_f}$, then any extension σ'_f of σ_f for all histories verifies $\mathcal{H}_{\sigma'_f}$ and has a strictly larger history domain. Otherwise, we can let $P = [\![\mu]\!]_{\approx_Y^{\mathsf{m}}}$ be the equivalence class for $\approx_Y^{\mathsf{m}}$ of a finite outcome $\mu \in \mathrm{Out}_f(\eta, \sigma_f)$ for some $\eta \in \Omega$, such that μ falls in case *(i)* or *(iii)* of $\mathcal{H}_{\sigma_f}$. Let $\mathrm{last}(P)$ be the non-empty set of states $\mathrm{last}(m_Y(\eta))$ with $\eta \in P$ ($\mathrm{last}(P)$ does not depend on the choice of the representatives $\eta \in P$ because P is an equivalence class for $\approx_Y^{\mathsf{m}}$). We build the ATL model $C_P = \langle \mathrm{Agt}_P, \mathrm{St}_P, \Pi_P, \pi_P, \mathrm{Act}_P, d_P, o_P\rangle$ where

- $\mathrm{Agt}_P = \mathrm{Agt}$,
- $\mathrm{St}_P = \mathrm{last}(P) \cup \{W, L\}$ where W and L are fresh states,
- $\Pi_P = \{win, last\}$,
- $\mathrm{Act}_P = \mathrm{Act} \cup \{loop\}$ where $loop$ is a fresh action,
- $\pi_P(W) = \{win\}$, $\pi_P(L) = \emptyset$, and $\pi_P(s) = \{last\}$ if $s \in \mathrm{last}(P)$,
- for $a \in \mathrm{Agt}$ and $s \in \mathrm{last}(P)$, $d_P(a, W) = d_P(a, L) = \{loop\}$, $d_P(a, s) = d(a, s) \cap A_a$ where $A_a = \bigcup_{\mu \in P \cap \Theta} \bigcup_{\kappa \in C(\mu)} \gamma(\kappa(a))$. The intuition is that agent a is allowed to do an action α iff $s = \mu[j]$ for an outcome $\mu \in \mathrm{Out}(\rho[i], \sigma)$ that did not satisfy ϕ_2 at index j and a may do α after $\mu_{\leq j}$ in $\mathcal{S}$.
- $o_P(W, loop, \ldots, loop) = W$, $o_P(L, loop, \ldots, loop) = L$, and, for $s \in \mathrm{last}(P)$ and $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n) \in d_P(1, s) \times \cdots \times d_P(n, s)$,

$$o_P(s, \alpha_1, \ldots, \alpha_n) = \begin{cases} s' & \text{if } \exists \eta \in P,\ \mathrm{last}(\eta) = s \text{ and } \eta\vec{\alpha}s' \in P, \\ W & \text{if } \exists \eta \in P,\ \mathrm{last}(\eta) = s, \text{ and } \eta\vec{\alpha}s' \in \Omega \setminus P, \\ L & \text{otherwise.} \end{cases}$$

We consider the ATL formula $\phi_{\mathsf{ATL}} = \langle Y\rangle_{\mathsf{ATL}} last\ \mathcal{U}\ win$ where $\langle\cdot\rangle_{\mathsf{ATL}}$ is the strategic operator in ATL.

Let us prove all states $s \in \{\mathrm{last}(\eta) \mid \eta \in P \cap \Omega \wedge (\eta, 1, \kappa) \not\models_f \phi_1 \mathcal{U}\, \phi_2\}$ verify $s \models_{\mathsf{ATL}} \phi_{\mathsf{ATL}}$, *i.e.*, prove that all last state s of a finite path η from case *(i)* or *(iii)* of $\mathcal{H}_{\sigma_f}$, such that $\eta \in P$, verifies $s \models_{\mathsf{ATL}} \phi_{\mathsf{ATL}}$. Let such a path η and state $s = \mathrm{last}(\eta)$ with $\eta \in P \cap \Omega$ and $(\eta, 1, \kappa) \not\models_f \phi_1 \mathcal{U} \phi_2$. We first show we can take $\mu \in \Theta \cap P$ and $j \in \mathbb{N}$ such that $\mu[j] = s$. Indeed, $\eta \in P \cap \Omega$ so there is $\mu \in \mathrm{Out}(\rho[i], \sigma)$ and $j \in \mathbb{N}$ such that $\mu_{\leq j} \approx_Y^{\mathsf{M}} \eta$. We can change the actions of agents in $\mathrm{Agt} \setminus Y$ in μ, and even so, we will have $\mu \in \mathrm{Out}(\rho[i], \sigma)$ and $\mu_{\leq j} \approx_Y^{\mathsf{M}} \eta$ (because outcomes contains all possible actions for agents outside the coalition Y). In particular, we can take actions of $\mathrm{Agt} \setminus Y$ such that $\mu_{\leq j} \approx_{\mathrm{Agt}}^{\mathsf{M}} \eta$, where the equivalence is for all agents. If $(\mu_{\leq j}, 1, \kappa) \models_f \phi_1 \mathcal{U}\, \phi_2$, then $(\eta, 1, \kappa) \models_f \phi_1 \mathcal{U}\, \phi_2$ too (by the similarity of $\mu_{\leq j}$ and η when $\mu_{\leq j} \approx_{\mathrm{Agt}}^{\mathsf{M}} \eta$), which was assumed false. So $(\mu_{\leq j}, 1, \kappa) \models_f \mathcal{G}(\phi_1 \wedge \neg\phi_2)$ (because $(\mu, 1, \kappa) \models \phi_1 \mathcal{U}\, \phi_2$). This proves that $\mu_{\leq j} \in P \cap \Theta$. For the rest of the proof, the symbols s, μ and j will only refer to these.

Let σ_{ATL} the strategy assignment such that, for all finite path η in C_P starting from s and $a \in Y$, if all states of η are in $\mathrm{last}(P)$ then $\sigma_{\mathsf{ATL}}(a)(\eta) = \sigma(a)(\mu_{\leq j} \cdot \eta)$, and, if η ends in L or M, $\sigma_{\mathsf{ATL}}(a)(\eta) = loop$. As $\mu_{\leq j} \in P \cap \Theta$, each action of an agent $a \in Y$ is in A_a, so the strategy σ_{ATL} is feasible in C_P. We claim that σ_{ATL} is winning for $s \models \phi_{\mathsf{ATL}}$. Let $\eta \in \mathrm{Out}_{\mathsf{ATL}}(s, \sigma_{\mathsf{ATL}})$, where $\mathrm{Out}_{\mathsf{ATL}}$ is defined as Definition 2.9 but disregards the capacities (it is like the outcomes in ATL but records the actions to form a path). There are three cases: *(i)* all states from η are in $\mathrm{last}(P)$, *(ii)* η reaches L, or *(iii)* η reaches W. We show that the two first cases are absurd.

Suppose, all states from η are in $\mathrm{last}(P)$. By the definition of m_Y, all the combinations of actions for agents in Y between two consecutive states of η appear in μ. As such, we can replace all transitions $q\vec{\alpha}q'$ from η by a transition $q\vec{\beta}q' \sim_Y q\vec{\alpha}q'$ (where $\sim_Y = \bigcap_{a \in Y} \sim_a$) that appears in μ. After replacement, η is still an outcome from $\mathrm{Out}_{\mathsf{ATL}}(s, \sigma_{\mathsf{ATL}})$ since we changed only $\mathrm{Agt} \setminus Y$'s actions. Moreover, the path $\mu \cdot \eta$ is in $\mathrm{Out}(\rho[i], \sigma)$ and it verifies $(\mu \cdot \eta, 1, \kappa) \models \mathcal{G}(\phi_1 \wedge \neg\phi_2)$ (because it stays forever in $\mathrm{last}(P)$ using previously used actions for all agents). This is absurd because it is an outcome of the winning strategy σ.

Suppose $\eta = s_1\vec{\alpha}_1 \ldots s_k\vec{\alpha}_k L \ldots$ reaches state L for the first time at index $k+1$ (so $s_k \in \mathrm{last}(P)$). By definition of o_P and d_P (in particular the definition of A_a), there is a finite path $\mu' \in P \cap \Theta$ such that $\vec{\alpha}_k$ is feasible by a complete capacity assignment $\kappa' \in C(\mu')$. Once again, we can modify actions of agents in $\mathrm{Agt} \setminus Y$ in $\eta_{\leq k}$ and μ' such that $\kappa' \in C(\mu') \cap C(\eta_{\leq k})$, and we let $\lambda = \mu' \cdot \eta_{\leq k}$. We have $\lambda \in P$ with $\mathrm{last}(\lambda) = s_k$. There is a unique state $s_{k+1} \in \mathrm{St}$ such that $\lambda\vec{\alpha}_k s_{k+1}$ is a path in $\mathcal{S}$. Defined as such, $\lambda\vec{\alpha}_k s_{k+1}$ is also the prefix of an outcome in $\mathrm{Out}(\rho[i], \sigma)$, and consequently, $\lambda\vec{\alpha}_k s_{k+1} \in \Omega$. If $\lambda\vec{\alpha}_k s_{k+1} \in P$, then λ should reach s_{k+1} instead of L, which is absurd. If, $\lambda\vec{\alpha}_k s_{k+1} \notin P$, according to Equation (4), η should reach W instead of L, which is also absurd.

Finally, η reaches W and $s \models_{\mathsf{ATL}} \phi_{\mathsf{ATL}}$, and this is true for all states $s \in \{\mathrm{last}(\eta) \mid \eta \in P \cap \Omega \wedge (\eta, 1, \kappa) \not\models_f \phi_1 \mathcal{U}\, \phi_2\}$.

A famous result about ATL is that having a winning strategy implies having a winning memoryless strategy for all states satisfying the formula [3]. We denote by σ'_{ATL} such a winning memoryless strategy. Let σ'_f denote the partial-history strategy assignment such that, for $a \in Y$ and $\eta \in \mathrm{Paths}^{<\omega}$, if $\eta \in \mathrm{dom}(\sigma_f(a))$ then $\sigma'_f(a)(\eta) = \sigma_f(a)(\eta)$ and, if $\eta \in P$, $\sigma'_f(a)(\eta) = \sigma'_{\mathsf{ATL}}(a)(\mathrm{last}(\eta))$. Notice that this second case never redefines a value of $\sigma_f(a)$ because partial-history strategy assignments assign strategy with the same history domain for each agent. Let $\eta \in \Omega$ and $\lambda \in \mathrm{Out}_f(\eta, \sigma'_f)$. In most cases, $\mathcal{H}_{\sigma'_f}$ results from $\mathcal{H}_{\sigma_f}$ directly, but the case where $\lambda_{\leq j} \in P$ for some j requires more attention. In the CGS C_P, agents in $\mathrm{Agt} \setminus Y$ have more possible actions than in the CapCGS $\mathcal{S}$ because they are not constrained by theirs capacities. Consequently, the guaranty to stay in $\mathrm{last}(P)$ until making a transition corresponding

to reaching W holds. This means that for some $k_1 \geq j$, we have $\lambda_{\leq k_1} \in \Omega$ with $[\![\lambda_{\leq k_1}]\!]_{\approx_Y^{\mathsf{m}}} > [\![\lambda_{\leq j}]\!]_{\approx_Y^{\mathsf{m}}}$. If σ_f's strategies were defined for $\lambda_{\leq k_1}$, either $(\lambda, 1, \kappa) \models_{(f)} \phi_1 \mathcal{U} \phi_2$ (and $\mathcal{H}_{\sigma'_f}$ is satisfied), or we can take $k_2 \geq k_1$ such that $[\![\lambda_{\leq k_2}]\!]_{\approx_Y^{\mathsf{m}}} > [\![\lambda_{\leq k_1}]\!]_{\approx_Y^{\mathsf{m}}}$, *etc.* As there is a bounded number of equivalence classes for $\approx_Y^{\mathsf{M}}$, at some point $(\lambda, 1, \kappa) \models_{(f)} \phi_1 \mathcal{U} \phi_2$ or $\lambda \in \Omega$ with $[\![\lambda]\!]_{\approx_Y^{\mathsf{m}}} > [\![\lambda_{\leq j}]\!]_{\approx_Y^{\mathsf{m}}}$. Finally, $\mathcal{H}_{\sigma'_f}$ holds with σ'_f defined for strictly more histories. By induction and the finiteness of the number of $\approx_Y^{\mathsf{M}}$'s equivalence classes, we can take the strategy assignment σ' (as the limit of the finite induction) defined for all histories. All outcomes $\eta \in \mathrm{Out}(\rho[i], \sigma')$ verify $(\eta, 1, \kappa) \models \phi_1 \mathcal{U} \phi_2$. Notice that the strategy σ' is only uniform for $\approx_Y^{\mathsf{M}}$ by definition. This means that agents remember also the actions of other agents of the coalition. However, we can prove that we can build a memory-bounded strategy assignment from σ' because all agents in Y can compute σ' and assume each other agent will follow the plan, and because a same transition $s\alpha s'$ is remembered $T(s\alpha s', a)$ times by each agent a according to Equation (1).

This finishes the proof for the case $(\rho, i, \kappa) \models \langle Y \rangle \phi_1 \mathcal{U} \phi_2$. The case $\langle Y \rangle \phi_1 \mathcal{R} \phi_2$ is similar except that the reduction into ATL uses the ATL formula $\phi'_{\mathsf{ATL}} = \langle Y \rangle_{\mathsf{ATL}} win \mathcal{R} last$. □

Given Theorem 4.4, we can reduce CapATL model checking to memoryless ATEL model checking, which implies the following.

Theorem 4.5 (CapATL decidability). *CapATL model-checking problem is decidable. The problem is in NEXPTIME.*

Proof. We proceed with a reduction to ATEL [31] model checking. We will use the memoryless semantic with uniform strategy space in ATEL, over CGS with imperfect information. Let $\mathcal{S} = \langle \mathrm{Agt}, \mathrm{Cap}, \mathrm{St}, \Pi, \pi, \mathrm{Act}, \Gamma, \gamma, d, o \rangle$ be a CapCGS and ϕ be a CapATL formula without nested strategic operators. We first build $C_{\mathcal{S}} = \langle \mathrm{Agt}_{\mathcal{S}}, \mathrm{St}_{\mathcal{S}}, \Pi_{\mathcal{S}}, \pi_{\mathcal{S}}, \mathrm{Act}_{\mathcal{S}}, d_{\mathcal{S}}, o_{\mathcal{S}}, \{\equiv_a\}_{a \in \mathrm{Agt}_{\mathcal{S}}} \rangle$, a CGS with imperfect information such that, given two finite paths $\rho, \eta \in \mathrm{Paths}^{<\omega}$,

- $\mathrm{Agt}_{\mathcal{S}} = \mathrm{Agt}$,
- $\mathrm{St}_{\mathcal{S}}$ is the (finite) set of equivalence classes of $\approx_{\mathrm{Agt}}^{\mathsf{M}}$,
- $\Pi_{\mathcal{S}}$ contains Π and a fresh proposition ℓ_κ for each complete capacity assignment κ.
- $\pi_{\mathcal{S}}([\![\rho]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}) = \pi(\mathrm{last}(\rho)) \cup \bigcup_{\kappa \in C(\rho)} \ell_\kappa$
- $\mathrm{Act}_{\mathcal{S}} = \mathrm{Act}$,
- for $a \in \mathrm{Agt}$, $d_{\mathcal{S}}(a, [\![\rho]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}) = d(a, \mathrm{last}(\rho)) \cap \bigcup_{\kappa \in C(\rho)} \gamma(\kappa(a))$,
- for $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n) \in d_{\mathcal{S}}(1, [\![\rho]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}) \times \cdots \times d_{\mathcal{S}}(n, [\![\rho]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}})$, we let the transition $o_{\mathcal{S}}([\![\rho]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}, \alpha_1, \ldots, \alpha_n) = [\![\eta]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$ iff $\rho \vec{\alpha} \mathrm{last}(\eta) \approx_{\mathrm{Agt}}^{\mathsf{M}} \eta$,
- for $a \in \mathrm{Agt}$, we set $[\![\rho]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}} \equiv_a [\![\eta]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$ iff $[\![\rho]\!]_{\approx_a^{\mathsf{M}}} = [\![\eta]\!]_{\approx_a^{\mathsf{M}}}$.

Let $\mathcal{K}_{\mathsf{ATEL}}$ and $\langle\cdot\rangle_{\mathsf{ATEL}}$ be the knowledge and strategic operators in ATEL. We transform ϕ into an ATEL formula ϕ_{ATEL} by replacing $\langle Y \rangle$ by $\langle Y \rangle_{\mathsf{ATEL}}$ and $\mathcal{K}^a_{\mathsf{cap}}(\varphi)$ by $\mathcal{K}^a_{\mathsf{ATEL}}(\varphi')$ where φ' is the conjunction of $(\neg \ell_\kappa)$ for all complete capacity assignment κ such that $\kappa \not\models \varphi$. By Theorem 4.4, we have $s \models \phi$ iff $[\![s]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}} \models_{\mathsf{ATEL}} \phi_{\mathsf{ATEL}}$.

The model-checking procedure is the following: *(i)* we build the ATEL model (in exponential time) that is exponentially larger than the initial CapCGS, *(ii)* for each subformula of the form $\phi' = \langle Y \rangle \psi$, starting from the innermost formulae, we compute the set of states

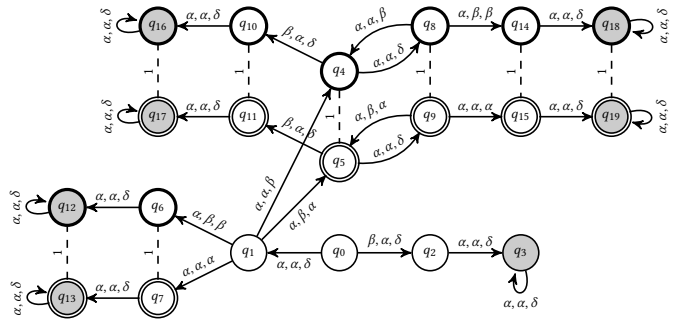


Figure 2: The CGS with imperfect information obtained from the model-checking transformation on the CapCGS from Figure 1.

$s \in \mathrm{St}$ such that $[\![s]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}} \models_{\mathsf{ATEL}} \phi'_{\mathsf{ATEL}}$, where ϕ'_{ATEL} is obtained as described above, *(iii)* we label each state s from the previous step with a fresh proposition $\ell_{\phi'}$ and replace the subformula ϕ' by ℓ_ϕ in the original formula ϕ, *(iv)* we iterate to step *(ii)* until the outermost formula, *(v)* we return the set of states that validate the outermost formula. In step *(ii)*, the states of the form $[\![s]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$ are distinguishable from all other states by all agents. So the question whether $[\![s]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}} \models_{\mathsf{ATEL}} \phi'_{\mathsf{ATEL}}$ should hold with the same strategy from states equivalent to $[\![s]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$ does not apply (*cf.*, [22]). The construction time and size of $C_{\mathcal{S}}$ are exponential with respect to $\mathcal{S}$ and memoryless ATEL model-checking without nested strategic operators is NP-complete [31] (but happens on an exponential size model). Step *(ii)* runs a polynomial number of time with respect to the formula length (once for each strategic operator). Finally, CapATL model checking is in NEXPTIME. □

Example 4.6. We display a simple example of the model-checking procedure based on the CapCGS of Figure 1 from Example 2.2. After transformation, we get the CGS with imperfect information from Figure 2. We display only the states generated by path starting in s_0. States linked with a dashed line are equivalent for $\approx_1$. Let κ_1 (resp. κ_2) be the complete capacity assignment such that $\kappa_1(3) = c_1$ (resp. $\kappa_2(3) = c_2$). Gray states have the atomic proposition *gray*, double states have ℓ_{κ_1}, thick states have ℓ_{κ_2}, and normal edges states have the last two. The state q_0 is $[\![s_0]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$ and the other states are the equivalence classes for the paths in $\mathcal{S}$ reaching them from q_0 with the same transitions. For instance, $q_1 = [\![s_0(\alpha, \alpha, \delta)s_1]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$ and $q_6 = [\![s_0(\alpha, \alpha, \delta)s_1(\alpha, \beta, \beta)s_0]\!]_{\approx_{\mathrm{Agt}}^{\mathsf{M}}}$. The formula $\phi = \langle 1 \rangle \mathcal{F}(gray \wedge \mathcal{K}^1_{\mathsf{cap}}(3))$ from Example 3.3 and Figure 1 turns into $\phi_{\mathsf{ATEL}} = \langle 1 \rangle_{\mathsf{ATEL}} \mathcal{F}(gray \wedge (\mathcal{K}^2_{\mathsf{ATEL}}(\ell_{\kappa_1} \wedge \neg \ell_{\kappa_2}) \vee \mathcal{K}^2_{\mathsf{ATEL}}(\ell_{\kappa_2} \wedge \neg \ell_{\kappa_1})))$. So, we have that $s_0 \models \phi$ holds because $q_0 \models_{\mathsf{ATEL}} \phi_{\mathsf{ATEL}}$ holds.

5 CASE STUDY

We consider a team of security engineers that wants to design a honeypot to identify an attacker in their company network. They assume three different types of attacker, denoted by att_1, att_2, and att_3, which differ because they master different vulnerabilities. It

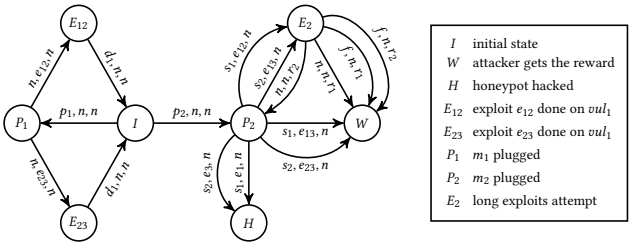


Figure 3: CapCGS for the case study.

is known that att_1 can do the exploits e_{12}, e_{13}, and e_1, while att_2 masters e_{12} and e_{23}, and att_3 knows e_{13}, e_{23}, and e_3. Precisely, the team designs two virtual machines, m_1 and m_2, with different vulnerabilities, namely vul_1 in m_1, and both vul_2 and vul_3 in m_2. They identify that vul_1 suffers from the exploits e_{12} and e_{23} that induce a different intermediary state (E_{12} and E_{23}, respectively) in the system. Moreover, depending on the service exposed on m_2, the vulnerability vul_2 can be directly exploited by e_{23} if the service is s_2 or e_{13} if the service is s_1. Vulnerability vul_2 also suffers from e_{12} with service s_1 and e_{13} with service s_2. However, in the two latter cases (called *long exploits*), the system goes in an intermediary state E_2 and the exploit success depends on the environment: in the random case r_1 the attack succeeds, and in the case r_2 the attack fails. To avoid this randomness, the engineers enable the honeypot to produce a fake output (denoted by f) to make the attacker think the attack succeeded, whatever the environment random choice r_1 or r_2. As mentioned, m_2 is also vulnerable to vul_3 if the service is s_1 (resp. s_2) and the attacker uses the advanced exploit e_1 (resp. e_3). The exploitation of vul_3 is dangerous because the attacker can compromise the honeypot and get root access to the host machine. Finally, the engineers decide to put a file of mock passwords in the machine m_2 accessible if the attacker exploits vul_2.

Thus, we can formalize the problem as a three-agent CapCGS where the first agent is the defender ($D = 1$), the second is the attacker ($A = 2$), and the third is the environment ($E = 3$). The defender has two capacities: $\Gamma(D) = \{real, honeypot\}$ representing a real system or a honeypot. The attacker has three capacities $\Gamma(A) = \{att_1, att_2, att_3\}$, one for each attacker profile, and the environment has a unique capacity $\Gamma(E) = \{env\}$. Here are the actions available to each capacity in this scenario. For the real defender, $\gamma(real) = \{p_1, p_2, d_1, d_2, s_1, s_2, n\}$: *i.e.*, plug (resp. disconnect) the machine m_1 with p_1 (resp. d_1) and m_2 with p_2 (resp. d_2), set service 1 with s_2 and 2 with s_2, or skip with n. The honeypot has one additional action f to produce a fake output and pretend the long exploits of vul_2 succeeded whatever the environment's randomness, so $\gamma(honeypot) = \{p_1, p_2, d_1, d_2, s_1, s_2, f, n\}$. Finally, $\gamma(att_1) = \{e_{12}, e_{13}, e_1, n\}$, $\gamma(att_2) = \{e_{12}, e_{23}, n\}$, $\gamma(att_3) = \{e_{13}, e_{23}, e_3, n\}$, and the environment has $\gamma(env) = \{r_1, r_2, n\}$. The agents, capacities, and actions are summarized in Tables 1 and 2. Moreover, the CapCGS in Figure 3 formalizes the system according to the description, where some loop transitions are ignored for clarity.

We want to ensure that the honeypot can detect the attacker profile, that the attacker does not determine that the system is not real, and that the honeypot is not compromised. The following formula can express the desired property, where the atomic proposition

Agent	Capacity	Actions
D	$honeypot$	$\{p_1, p_2, d_1, d_2, s_1, s_2, f, n\}$
	$real$	$\{p_1, p_2, d_1, d_2, s_1, s_2, n\}$
A	att_1	$\{e_{12}, e_{13}, e_1, n\}$
	att_2	$\{e_{12}, e_{23}, n\}$
	att_3	$\{e_{13}, e_{23}, e_3, n\}$
E	env	$\{r_1, r_2, n\}$

Table 1: Agents, capacities, and actions for the case study.

Action	Description
p_1	plug machine m_1
p_2	plug machine m_2
d_1	disconnect machine m_1
d_2	disconnect machine m_2
s_1	use service s_1 on m_2
s_2	use service s_2 on m_2
f	fake long exploits success of vul_2
e_1	exploit e_1 against vul_3
e_3	exploit e_3 against vul_3
e_{12}	exploit e_{12} against vul_1 or vul_2
e_{13}	exploit e_{13} against vul_1 or vul_2
e_{23}	exploit e_{23} against vul_1 or vul_2
r_1	random environment choice r_1
r_2	random environment choice r_2
n	do nothing

Table 2: Actions description for the case study.

hacked holds only in H and *win* holds only in W:

$$\langle D \mapsto honeypot\rangle(\neg hacked)\mathcal{U}(\mathcal{M}^A_{\text{cap}}(D \mapsto real) \wedge win \wedge \mathcal{K}^D_{\text{cap}}(A))$$

It turns out that state I verifies this property in the model. The winning strategy for the defender is to first use p_1 in I, then use p_2 the second time and s_1 in P_2 if the computation was in E_{23} and s_2 if the computation was in E_{12}. Finally, the defender can use f in E_2, and still the attacker cannot be sure the defender is a honeypot because the environment might have done r_1.

6 CONCLUSION

This paper introduces CapATL to reason about MAS where agents are bounded to capacities that restrict their possible actions. The capacities account for the diversity of entities that may play the role of an agent. We proved the decidability of CapATL model checking in NEXPTIME. A cyber deception use case showed CapATL applicability and the practical properties to design and pilot an adaptable honeypot, considering the uncertain profiles of the agents.

There are several directions to continue working on CapATL. We would like to investigate different approximations and restrictions to find an efficient model-checking algorithm, satisfactory for real-world applications. We plan to look at quantitative aspects, such as in [4, 25–27], but for capacities. Moreover, we can extend our idea to SL [6, 7, 28]. Finally, we want to investigate further the theory of capacities too, for including them in a powerful language.

ACKNOWLEDGMENTS

This work was carried out within SEIDO Lab, a joint research laboratory covering research topics in the field of smart grids, *e.g.*, distributed intelligence, service collaboration, cybersecurity, and privacy. It involves researchers from academia (Télécom Paris, Télécom SudParis, CNRS LAAS) and industry (EDF R&D).

REFERENCES

[1] Thomas Ågotnes. 2006. Action and Knowledge in Alternating-Time Temporal Logic. *Synth.* 149, 2 (2006), 375–407. https://doi.org/10.1007/S11229-005-3875-8

[2] Natasha Alechina, Brian Logan, Nguyen Hoang Nga, and Abdur Rakib. 2010. Resource-Bounded Alternating-Time Temporal Logic. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1* (Toronto, Canada) *(AAMAS '10)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 481–488.

[3] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-Time Temporal Logic. *Journal of the ACM* 49, 5 (Sept. 2002), 672–713. https://doi.org/10.1145/585265.585270

[4] Benjamin Aminof, Vadim Malvone, Aniello Murano, and Sasha Rubin. 2018. Graded Modalities in Strategy Logic. *Information and Computation* 261 (2018), 634–649. https://doi.org/10.1016/j.ic.2018.02.022

[5] Francesco Belardinelli, Angelo Ferrando, and Vadim Malvone. 2023. An Abstraction-Refinement Framework for Verifying Strategic Properties in Multi-Agent Systems with Imperfect Information. *Artificial Intelligence* 316 (March 2023), 103847. https://doi.org/10.1016/j.artint.2022.103847

[6] Francesco Belardinelli, Angelo Ferrando, Wojciech Jamroga, Vadim Malvone, and Aniello Murano. 2023. Scalable Verification of Strategy Logic through Three-Valued Abstraction. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, ijcai.org, Macao, China, 46–54. https://doi.org/10.24963/ijcai.2023/6

[7] Francesco Belardinelli, Wojciech Jamroga, Damian Kurpiewski, Vadim Malvone, and Aniello Murano. 2019. Strategy Logic with Simple Goals: Tractable Reasoning about Strategies. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, Macao, China, 88–94. https://doi.org/10.24963/ijcai.2019/13

[8] Francesco Belardinelli, Alessio Lomuscio, Vadim Malvone, and Emily Yu. 2022. Approximating Perfect Recall When Model Checking Strategic Abilities: Theory and Applications. *J. Artif. Intell. Res.* 73 (2022), 897–932. https://doi.org/10.1613/jair.1.12539

[9] Raphaël Berthon, Bastien Maubert, and Aniello Murano. 2017. Decidability Results for ATL* with Imperfect Information and Perfect Recall. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (São Paulo, Brazil) *(AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1250–1258.

[10] Thomas Brihaye, Véronique Bruyere, and Jean-François Raskin. 2005. On Optimal Timed Strategies. In *International Conference on Formal Modeling and Analysis of Timed Systems (SpringerLink)*, Paul Pettersson and Wang Yi (Eds.). Springer, Springer Berlin Heidelberg, Berlin, Heidelberg, 49–64.

[11] Taolue Chen and Jian Lu. 2007. Probabilistic Alternating-Time Temporal Logic and Model Checking Algorithm, In Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007). *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)* 2, 35–39. https://doi.org/10.1109/FSKD.2007.458

[12] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, Roderick Bloem, et al. 2018. *Handbook of Model Checking*. Springer eBook Collection, Vol. 10. Springer Cham, Cham. https://doi.org/10.1007/978-3-319-10575-8 Includes bibliographical references.

[13] Alexandre David, Peter G. Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Sørensen, and Jakob H. Taankvist. 2014. On Time with Minimal Expected Cost!. In *Automated Technology for Verification and Analysis*, Franck Cassez and Jean-François Raskin (Eds.). Springer, Cham, 129–145.

[14] Catalin Dima and Ferucio Laurentiu Tiplea. 2011. Model-Checking ATL under Imperfect Information and Perfect Recall Semantics Is Undecidable. *CoRR* abs/1102.4225 (2011). https://hal.science/hal-01699948

[15] Angelo Ferrando and Vadim Malvone. 2023. Towards the Verification of Strategic Properties in Multi-Agent Systems with Imperfect Information. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023*, Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh (Eds.). ACM, 793–801. https://doi.org/10.5555/3545946.3598713

[16] Andreas Herzig, Emiliano Lorini, and Dirk Walther. 2013. Reasoning about Actions Meets Strategic Logics. In *Logic, Rationality, and Interaction*, Davide Grossi, Olivier Roy, and Huaxin Huang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 162–175.

[17] Xiaowei Huang, Kaile Su, and Chenyi Zhang. 2012. Probabilistic Alternating-Time Temporal Logic of Incomplete Information and Synchronous Perfect Recall, In Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada, Jörg Hoffmann and Bart Selman (Eds.). *Proceedings of the AAAI Conference on Artificial Intelligence* 26, 1, 765–771. https://doi.org/10.1609/aaai.v26i1.8214

[18] Wojciech Jamroga. 2003. Some Remarks on Alternating Temporal Epistemic Logic, In Formal Approaches to Multi-Agent Systems. *Formal Approaches to Multi-Agent Systems*. https://www.semanticscholar.org/paper/37e42747212142ef57bb4b36dcd12920225765b1

[19] Wojciech Jamroga, Michał Knapik, Damian Kurpiewski, and Łukasz Mikulski. 2019. Approximate Verification of Strategic Abilities under Imperfect Information. *Artificial Intelligence* 277 (2019), 103172. https://doi.org/10.1016/j.artint.2019.103172

[20] Wojciech Jamroga, Vadim Malvone, and Aniello Murano. 2019. Natural strategic ability. *Artif. Intell.* 277 (2019). https://doi.org/10.1016/j.artint.2019.103170

[21] Wojciech Jamroga, Vadim Malvone, and Aniello Murano. 2019. Natural Strategic Ability under Imperfect Information. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, Montreal, QC, Canada, 962–970. http://dl.acm.org/citation.cfm?id=3331791

[22] Wojciech Jamroga and Wiebe van der Hoek. 2004. Agents That Know How to Play. *Fundamenta Informaticae* 63 (2004), 185–219. 2-3.

[23] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. 2008. On the Expressiveness and Complexity of ATL. *Logical Methods in Computer Science* 4, 2 (May 2008). https://doi.org/10.2168/LMCS-4(2:7)2008

[24] Yves Lespérance, Hector J. Levesque, Fangzhen Lin, and Richard B. Scherl. 2000. Ability and Knowing How in the Situation Calculus. *Studia Logica* 66, 1 (2000), 165–186. https://doi.org/10.1023/A:1026761331498

[25] Vadim Malvone, Fabio Mogavero, Aniello Murano, and Loredana Sorrentino. 2018. Reasoning about graded strategy quantifiers. *Information and Computation* 259 (2018), 390–411. https://doi.org/10.1016/j.ic.2017.08.010

[26] Vadim Malvone and Aniello Murano. 2017. Reasoning About Additional Winning Strategies in Two-Player Games. In *Multi-Agent Systems and Agreement Technologies - 15th European Conference, EUMAS 2017, and 5th International Conference, AT 2017, Évry, France, December 14-15, 2017, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10767)*, Francesco Belardinelli and Estefania Argente (Eds.). Springer, 163–171. https://doi.org/10.1007/978-3-030-01713-2_12

[27] Vadim Malvone, Aniello Murano, and Loredana Sorrentino. 2018. Additional Winning Strategies in Reachability Games. *Fundam. Informaticae* 159, 1-2 (2018), 175–195. https://doi.org/10.3233/FI-2018-1662

[28] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. 2014. Reasoning about Strategies: On the Model-Checking Problem. *ACM Transactions on Computational Logic* 15, 4 (Aug. 2014), 1–47. https://doi.org/10.1145/2631917

[29] Hoang Nga Nguyen and Abdur Rakib. 2019. Probabilistic Resource-Bounded Alternating-Time Temporal Logic. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) *(AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2141–2143.

[30] Henning Schnoor. 2010. Strategic Planning for Probabilistic Games with Incomplete Information. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1* (Toronto, Canada) *(AAMAS '10)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1057–1064.

[31] Pierre-Yves Schobbens. 2004. Alternating-Time Logic with Imperfect Recall. *Electronic Notes in Theoretical Computer Science* 85, 2 (2004), 82–93. https://doi.org/10.1016/S1571-0661(05)82604-0 LCMAS 2003, Logic and Communication in Multi-Agent Systems.

[32] Wiebe van der Hoek, B. van Linder, and John-Jules Ch. Meyer. 1994. A Logic of Capabilities. In *Logical Foundations of Computer Science*, Anil Nerode and Yu. V. Matiyasevich (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 366–378.

[33] Wiebe van der Hoek and Michael J. Wooldridge. 2002. Tractable Multiagent Planning for Epistemic Goals. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems*. ACM, Bologna, Italy, 1167–1174. https://doi.org/10.1145/545056.545095

[34] Wiebe van der Hoek and Michael J. Wooldridge. 2003. Cooperation, Knowledge, and Time: Alternating-Time Temporal Epistemic Logic and Its Applications. *Stud Logica* 75, 1 (2003), 125–157. https://doi.org/10.1023/A:1026185103185

[35] Dirk Walther, Wiebe van der Hoek, and Michael Wooldridge. 2007. Alternating-Time Temporal Logic with Explicit Strategies. In *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge* (Brussels, Belgium) *(TARK '07)*. Association for Computing Machinery, New York, NY, USA, 269–278. https://doi.org/10.1145/1324249.1324285

[36] Thomas Ågotnes, Valentin Goranko, and Wojciech Jamroga. 2007. Alternating-Time Temporal Logics with Irrevocable Strategies. In *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge* (Brussels, Belgium) *(TARK '07)*. Association for Computing Machinery, New York, NY, USA, 15–24. https://doi.org/10.1145/1324249.1324256